Computing Igusa's local zeta function of univariates in deterministic polynomial-time

Ashish Dwivedi
(IIT Kanpur)

joint work
with
Nitin Saxena
(IIT Kanpur)

ANTS 2020

14th Algorithmic
Number Theory Symposium

Introduction

Introduction

We are interested in computing certain function in this talk.

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Eg An exponential function: $f(x,y)= 2^x + 3^y$ then $f(1,3) = 29$.

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Eg An exponential function: $f(x,y)= 2^x + 3^y$ then $f(1,3) = 29$.

Or A polynomial function: $f(x,y) = x^2 + xy + 1$ then $f(1,-2) = 0$.

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Eg An exponential function: $f(x,y)= 2^x + 3^y$ then $f(1,3) = 29$.

Or A polynomial function: $f(x,y) = x^2 + xy + 1$ then $f(1,-2) = 0$.

But these functions are nice explicit functions with finite description.

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Eg An exponential function: $f(x,y)= 2^x + 3^y$ then $f(1,3) = 29$.

Or A polynomial function: $f(x,y) = x^2 + xy + 1$ then $f(1,-2) = 0$.

But these functions are nice explicit functions with finite description.

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Eg An exponential function: $f(x,y)= 2^x + 3^y$ then $f(1,3) = 29$.

Or A polynomial function: $f(x,y) = x^2 + xy + 1$ then $f(1,-2) = 0$.

But these functions are nice explicit functions with finite description.

A generating function?

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Eg An exponential function: $f(x,y)= 2^x + 3^y$ then $f(1,3) = 29$.

Or A polynomial function: $f(x,y) = x^2 + xy + 1$ then $f(1,-2) = 0$.

But these functions are nice explicit functions with finite description.

A generating function?

Eg, $f(t) = 1 + t + t^2 + \ldots$ encoding the sequence $1, 1, 1, 1, 1, 1, \ldots$

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Eg An exponential function: $f(x,y)= 2^x + 3^y$ then $f(1,3) = 29$.

Or A polynomial function: $f(x,y) = x^2 + xy + 1$ then $f(1,-2) = 0$.

But these functions are nice explicit functions with finite description.

A generating function?

Eg, $f(t) = 1 + t + t^2 + \ldots$ encoding the sequence $1, 1, 1, 1, 1, 1, \ldots$

Or $g(t) = 1 + 2t + 3t^2 + \ldots$ encoding the sequence $1, 2, 3, 4, 5, \ldots$

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Eg An exponential function: $f(x,y) = 2^x + 3^y$ then $f(1,3) = 29$.

Or A polynomial function: $f(x,y) = x^2 + xy + 1$ then $f(1,-2) = 0$.

But these functions are nice explicit functions with finite description.

A generating function?

Eg, $f(t) = 1 + t + t^2 + \dots$ encoding the sequence $1, 1, 1, 1, 1, 1, \dots$

Or $g(t) = 1 + 2t + 3t^2 + \dots$ encoding the sequence $1, 2, 3, 4, 5, \dots$

In such cases, it is interesting to know the rational form of the function

Introduction

We are interested in computing certain function in this talk.

But, what do we really mean by computing a function?

Often it means evaluating the function at certain points.

Eg An exponential function: $f(x,y)= 2^x + 3^y$ then $f(1,3) = 29$.

Or A polynomial function: $f(x,y) = x^2 + xy + 1$ then $f(1,-2) = 0$.

But these functions are nice explicit functions with finite description.

A generating function?

Eg, $f(t) = 1 + t + t^2 + \ldots$ encoding the sequence $1, 1, 1, 1, 1, 1, \ldots$

Or $g(t) = 1 + 2t + 3t^2 + \ldots$ encoding the sequence $1, 2, 3, 4, 5, \ldots$

In such cases, it is interesting to know the rational form of the function

$f(t)= 1/(1-t)$ for $|t|<1$ and $g(t)= f(t)^2$.

Introduction

Introduction

Functions encoding other functions:

Introduction

Functions encoding other functions:

A generating function with coefficient as other functions!

Introduction

Functions encoding other functions:

A generating function with coefficient as other functions!

$$f(x,t) = g_0(x) + g_1(x)t + g_2(x)t^2 + \dots$$

Introduction

Functions encoding other functions:

A generating function with coefficient as other functions!

$$f(x,t) = g_0(x) + g_1(x)t + g_2(x)t^2 + \ldots$$

Fns $1/(1-t)$ and $1/(1-t)^2$ are rational forms of two simple cases of this function.

Introduction

Functions encoding other functions:

A generating function with coefficient as other functions!

$$f(x,t) = g_0(x) + g_1(x)t + g_2(x)t^2 + \ldots$$

Fns $1/(1-t)$ and $1/(1-t)^2$ are rational forms of two simple cases of this function.

Computing rational form becomes highly non-trivial when g is implicit.

Introduction

Functions encoding other functions:

A generating function with coefficient as other functions!

$$f(x,t) = g_0(x) + g_1(x)t + g_2(x)t^2 + \dots$$

Fns $1/(1-t)$ and $1/(1-t)^2$ are rational forms of two simple cases of this function.

Computing rational form becomes highly non-trivial when g is implicit.

An interesting example is Poincare series which encodes roots of a polynomial mod prime-powers:

Introduction

Functions encoding other functions:

A generating function with coefficient as other functions!

$$f(x,t) = g_0(x) + g_1(x)t + g_2(x)t^2 + \ldots$$

Fns $1/(1-t)$ and $1/(1-t)^2$ are rational forms of two simple cases of this function.

Computing rational form becomes highly non-trivial when g is implicit.

An interesting example is Poincare series which encodes roots of a polynomial mod prime-powers:

$$P_{f,p}(t) := \sum_{k=0}^{\infty} \frac{N_k(f)}{p^{nk}}\, t^k$$

Introduction

Functions encoding other functions:

A generating function with coefficient as other functions!

$$f(x,t) = g_0(x) + g_1(x)t + g_2(x)t^2 + \ldots$$

Fns $1/(1-t)$ and $1/(1-t)^2$ are rational forms of two simple cases of this function.

Computing rational form becomes highly non-trivial when g is implicit.

An interesting example is Poincare series which encodes roots of a polynomial mod prime-powers:

$$\boxed{P_{f,p}(t) := \sum_{k=0}^{\infty} \frac{N_k(f)}{p^{nk}}\, t^k}$$

f is polynomial in n variables with integer coefficients,

Introduction

Functions encoding other functions:

A generating function with coefficient as other functions!

$$f(x,t) = g_0(x) + g_1(x)t + g_2(x)t^2 + \ldots$$

Fns $1/(1-t)$ and $1/(1-t)^2$ are rational forms of two simple cases of this function.

Computing rational form becomes highly non-trivial when g is implicit.

An interesting example is Poincare series which encodes roots of a polynomial mod prime-powers:

$$P_{f,p}(t) := \sum_{k=0}^{\infty} \frac{N_k(f)}{p^{nk}}\, t^k$$

f is polynomial in n variables with integer coefficients,
p is a prime and $N_k(f)$ = number of zeros of f modulo p^k.

Introduction

Functions encoding other functions:

A generating function with coefficient as other functions!

$$f(x,t) = g_0(x) + g_1(x)t + g_2(x)t^2 + \ldots$$

Fns $1/(1-t)$ and $1/(1-t)^2$ are rational forms of two simple cases of this function.

Computing rational form becomes highly non-trivial when g is implicit.

An interesting example is Poincare series which encodes roots of a polynomial mod prime-powers:

$$P_{f,p}(t) := \sum_{k=0}^{\infty} \frac{N_k(f)}{p^{nk}} t^k$$

f is polynomial in n variables with integer coefficients,
p is a prime and $N_k(f)$ = number of zeros of f modulo p^k.

We will be interested in computing rational form of this function in this talk!

Zeta Functions

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Very likely these zeta fns posses special analytic and algebraic properties.

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Very likely these zeta fns posses special analytic and algebraic properties.

Which reveals striking information about the encoded objects (hidden otherwise).

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Very likely these zeta fns posses special analytic and algebraic properties.

Which reveals striking information about the encoded objects (hidden otherwise).

This makes them extremely interesting and often drives whole new area of maths.

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Very likely these zeta fns posses special analytic and algebraic properties.

Which reveals striking information about the encoded objects (hidden otherwise).

This makes them extremely interesting and often drives whole new area of maths.

Applications (everywhere): Computer science, Crypto, Number theory, Data science,

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Very likely these zeta fns posses special analytic and algebraic properties.

Which reveals striking information about the encoded objects (hidden otherwise).

This makes them extremely interesting and often drives whole new area of maths.

Applications (everywhere): Computer science, Crypto, Number theory, Data science, **Black-holes**.

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Very likely these zeta fns posses special analytic and algebraic properties.

Which reveals striking information about the encoded objects (hidden otherwise).

This makes them extremely interesting and often drives whole new area of maths.

Applications (everywhere): Computer science, Crypto, Number theory, Data science, **Black-holes**.

Eg, The very first zeta function— Euler-Riemann [1859] zeta function.

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Very likely these zeta fns posses special analytic and algebraic properties.

Which reveals striking information about the encoded objects (hidden otherwise).

This makes them extremely interesting and often drives whole new area of maths.

Applications (everywhere): Computer science, Crypto, Number theory, Data science, **Black-holes**.

Eg, The very first zeta function— Euler-Riemann [1859] zeta function.

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Very likely these zeta fns posses special analytic and algebraic properties.

Which reveals striking information about the encoded objects (hidden otherwise).

This makes them extremely interesting and often drives whole new area of maths.

Applications (everywhere): Computer science, Crypto, Number theory, Data science, **Black-holes**.

Eg, The very first zeta function— Euler-Riemann [1859] zeta function.

where s is a complex number with Re(s)>1.

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

Zeta Functions

Zeta functions are one of the most important class of functions which encode the counts of objects encompassing some mathematical structure.

Very likely these zeta fns posses special analytic and algebraic properties.

Which reveals striking information about the encoded objects (hidden otherwise).

This makes them extremely interesting and often drives whole new area of maths.

Applications (everywhere): Computer science, Crypto, Number theory, Data science, **Black-holes**.

Eg, The very first zeta function— Euler-Riemann [1859] zeta function.

where s is a complex number with Re(s)>1.

encodes the distribution of prime numbers.

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

Igusa Zeta Function

Igusa Zeta Function

Other zeta functions can be defined

Igusa Zeta Function

Other zeta functions can be defined Local

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Count of zeros of system of polynomial equations in finite fields of char p.

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

$\boxed{\text{Hasse-Weil Zeta Fn:}}$ Count of zeros of system of polynomial equations in finite fields of char p.

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

$\boxed{\text{Hasse-Weil Zeta Fn:}}$ Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Hasse-Weil Zeta Fn:

Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Given an n-variate polynomial f with integer coefficients and a prime p.

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

$\boxed{\text{Hasse-Weil Zeta Fn:}}$ Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Given an n-variate polynomial f with integer coefficients and a prime p.

Encodes count of zeros of polynomial f modulo prime powers p^k.

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Hasse-Weil Zeta Fn: Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Given an n-variate polynomial f with integer coefficients and a prime p.

Encodes count of zeros of polynomial f modulo prime powers p^k.

$$Z_{f,p}(s) = \int_{\mathbb{Z}_p^n} |f(\mathbf{x})|_p^s \, |d\mathbf{x}|$$

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Hasse-Weil Zeta Fn: Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Given an n-variate polynomial f with integer coefficients and a prime p.

Encodes count of zeros of polynomial f modulo prime powers p^k.

$$Z_{f,p}(s) = \int_{\mathbb{Z}_p^n} |f(\mathbf{x})|_p^s \, |d\mathbf{x}|$$

$\mathbb{Z}_p$ p-adic integers

s is complex

Re(s)>0

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Hasse-Weil Zeta Fn: Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Given an n-variate polynomial f with integer coefficients and a prime p.

Encodes count of zeros of polynomial f modulo prime powers p^k.

$$Z_{f,p}(s) = \int_{\mathbb{Z}_p^n} |f(\mathbf{x})|_p^s \, |d\mathbf{x}|$$

$\mathbb{Z}_p$ p-adic integers

s is complex

Re(s)>0

The original definition is not important

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Hasse-Weil Zeta Fn: Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Given an n-variate polynomial f with integer coefficients and a prime p.

Encodes count of zeros of polynomial f modulo prime powers p^k.

$$Z_{f,p}(s) = \int_{\mathbb{Z}_p^n} |f(\mathbf{x})|_p^s \, |d\mathbf{x}|$$

$\mathbb{Z}_p$ p-adic integers

s is complex

Re(s)>0

The original definition is not important $Z_{f,p}(s)$ is a **rational** function [Igusa'70s]

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Hasse-Weil Zeta Fn: Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Given an n-variate polynomial f with integer coefficients and a prime p.

Encodes count of zeros of polynomial f modulo prime powers p^k.

$$Z_{f,p}(s) = \int_{\mathbb{Z}_p^n} |f(\mathbf{x})|_p^s \, |d\mathbf{x}|$$

$\mathbb{Z}_p$ p-adic integers

s is complex

Re(s)>0

The original definition is not important $Z_{f,p}(s)$ is a **rational** function [Igusa'70s]

Poincare series $$P_{f,p}(t) := \sum_{k=0}^{\infty} \frac{N_k(f)}{p^{nk}} t^k$$

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Hasse-Weil Zeta Fn: Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Given an n-variate polynomial f with integer coefficients and a prime p.

Encodes count of zeros of polynomial f modulo prime powers p^k.

$$Z_{f,p}(s) = \int_{\mathbb{Z}_p^n} |f(\mathbf{x})|_p^s \, |d\mathbf{x}|$$

$\mathbb{Z}_p$ p-adic integers

s is complex

Re(s)>0

The original definition is not important

$Z_{f,p}(s)$ is a **rational** function [Igusa'70s]

Poincare series $$P_{f,p}(t) := \sum_{k=0}^{\infty} \frac{N_k(f)}{p^{nk}} t^k$$

$$P_{f,p}(t) = \frac{1 - t \, . \, Z_{f,p}(s)}{1 - t} \; ; \qquad t = p^{-s}$$

Igusa Zeta Function

Other zeta functions can be defined Local associated to a particular prime p.

Hasse-Weil Zeta Fn: Count of zeros of system of polynomial equations in finite fields of char p.

Igusa's local zeta function:

Given an n-variate polynomial f with integer coefficients and a prime p.

Encodes count of zeros of polynomial f modulo prime powers p^k.

$$Z_{f,p}(s) = \int_{\mathbb{Z}_p^n} |f(\mathbf{x})|_p^s \, |d\mathbf{x}|$$

$\mathbb{Z}_p$ p-adic integers

s is complex

Re(s)>0

The original definition is not important

$Z_{f,p}(s)$ is a **rational** function [Igusa'70s]

Poincare series
$$P_{f,p}(t) := \sum_{k=0}^{\infty} \frac{N_k(f)}{p^{nk}} t^k$$

$$P_{f,p}(t) = \frac{1 - t \, . \, Z_{f,p}(s)}{1 - t} \; ; \qquad t = p^{-s}$$

Computing $Z_{f,p}(s)$ boils down to compute rational form of poincare series $P(t)$.

The Problem

The Problem

Various methods developed to compute IZF for special family of polynomials.

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula Newton polygon method Resolution of Singularities

Not much said about their algorithmic aspect. Known methods are impractical.

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.   Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.   Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

Univariate Polynomials:

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.   Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

Univariate Polynomials: Better algorithms?

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.   Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

Univariate Polynomials: Better algorithms?   Simple rationality proof?

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.   Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

Univariate Polynomials: Better algorithms?   Simple rationality proof?

Input: Given 1-variable polynomial $f(x)$ with integer coefficients and a prime p.

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.   Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

Univariate Polynomials: Better algorithms?   Simple rationality proof?

Input: Given 1-variable polynomial $f(x)$ with integer coefficients and a prime p.

Output: Efficiently compute the associated Poincare series $P(t)=A(t)/B(t)$ for univariate polynomials $A(t)$ and $B(t)$ over rationals and $t = p^{-s}$.

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.   Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

Univariate Polynomials: Better algorithms?   Simple rationality proof?

Input: Given 1-variable polynomial $f(x)$ with integer coefficients and a prime p.

Output: Efficiently compute the associated Poincare series $P(t)=A(t)/B(t)$ for univariate polynomials $A(t)$ and $B(t)$ over rationals and $t = p^{-s}$.

Related to univariate **root-counting** mod p^k.

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.   Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

Univariate Polynomials: Better algorithms?   Simple rationality proof?

Input: Given 1-variable polynomial $f(x)$ with integer coefficients and a prime p.

Output: Efficiently compute the associated Poincare series $P(t)=A(t)/B(t)$ for univariate polynomials $A(t)$ and $B(t)$ over rationals and $t = p^{-s}$.

Related to univariate **root-counting** mod p^k.   Coding theory, Crypto, Factoring etc.

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula   Newton polygon method   Resolution of Singularities

Not much said about their algorithmic aspect.   Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

Univariate Polynomials: Better algorithms?   Simple rationality proof?

Input: Given 1-variable polynomial $f(x)$ with integer coefficients and a prime p.

Output: Efficiently compute the associated Poincare series $P(t)=A(t)/B(t)$ for univariate polynomials $A(t)$ and $B(t)$ over rationals and $t = p^{-s}$.

Related to univariate **root-counting** mod p^k.   Coding theory, Crypto, Factoring etc.

Recently [DMS19] computed $N_k(f)$ in det poly-time.

The Problem

Various methods developed to compute IZF for special family of polynomials.

Stationary Phase Formula Newton polygon method Resolution of Singularities

Not much said about their algorithmic aspect. Known methods are impractical.

Indeed, counting roots of n-variable poly in finite fields is already NP-Hard.

Univariate Polynomials: Better algorithms? Simple rationality proof?

Input: Given 1-variable polynomial $f(x)$ with integer coefficients and a prime p.

Output: Efficiently compute the associated Poincare series $P(t)=A(t)/B(t)$ for univariate polynomials $A(t)$ and $B(t)$ over rationals and $t = p^{-s}$.

Related to univariate **root-counting** mod p^k. Coding theory, Crypto, Factoring etc.

Recently [DMS19] computed $N_k(f)$ in det poly-time.

Question: Can we find expression for $N_k(f)$ if it exists?

Our Results

Our Results

Given an integral univariate $f(x)$

Our Results

Given an integral univariate $f(x)$ degree $\leq d$

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$\boxed{P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty}$$

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$\boxed{P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \; + \cdots \; \infty}$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$\boxed{P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty}$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

$$P(t) = \frac{A(t)}{B(t)};\ t = p^{-s}$$

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

$$P(t) = \frac{A(t)}{B(t)}; \; t = p^{-s}$$

$$\deg(A) = \tilde{O}(d^2)$$
$$\deg(B) = O(d)$$

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

$$P(t) = \frac{A(t)}{B(t)};\ t = p^{-s}$$

$$\deg(A) = \tilde{O}(d^2)$$
$$\deg(B) = O(d)$$

We also give the proof of rationality of P(t) by first principles.

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p} t + \frac{N_2(f)}{p^2} t^2 + \cdots \infty$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

$$P(t) = \frac{A(t)}{B(t)}; \; t = p^{-s}$$

$$\deg(A) = \tilde{O}(d^2)$$
$$\deg(B) = O(d)$$

We also give the proof of rationality of P(t) by first principles.

Zuniga-Galindo'03 gave a det poly-time algo

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

$$P(t) = \frac{A(t)}{B(t)};\ t = p^{-s}$$

$$\deg(A) = \tilde{O}(d^2)$$
$$\deg(B) = O(d)$$

We also give the proof of rationality of P(t) by first principles.

Zuniga-Galindo'03 gave a det poly-time algo when f completely splits over rationals

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p} t + \frac{N_2(f)}{p^2} t^2 + \cdots \infty$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

$$P(t) = \frac{A(t)}{B(t)}; \; t = p^{-s}$$

$$\deg(A) = \tilde{O}(d^2)$$
$$\deg(B) = O(d)$$

We also give the proof of rationality of P(t) by first principles.

Zuniga-Galindo'03 gave a det poly-time algo when f completely splits over rationals

Our algo makes no such assumptions

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

$$P(t) = \frac{A(t)}{B(t)};\ t = p^{-s}$$

$$\deg(A) = \tilde{O}(d^2)$$
$$\deg(B) = O(d)$$

We also give the proof of rationality of P(t) by first principles.

Zuniga-Galindo'03 gave a det poly-time algo when f completely splits over rationals

Our algo makes no such assumptions also works for f is defined over $\mathbb{Z}_p$.

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

$$P(t) = \frac{A(t)}{B(t)};\ t = p^{-s}$$

$$\deg(A) = \tilde{O}(d^2)$$
$$\deg(B) = O(d)$$

We also give the proof of rationality of P(t) by first principles.

Zuniga-Galindo'03 gave a det poly-time algo when f completely splits over rationals

Our algo makes no such assumptions also works for f is defined over $\mathbb{Z}_p$.

A closed form expression for $N_k(f)$ for univariate f.

Our Results

Given an integral univariate $f(x)$ degree $\leq d$ coeff. at most C prime p

We give first deterministic poly-time algorithm to compute associated Poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\,t + \frac{N_2(f)}{p^2}\,t^2 + \cdots \infty$$

where $N_k(f)$ is the number of zeros of $f(x)$ mod p^k with,

$$P(t) = \frac{A(t)}{B(t)};\ t = p^{-s}$$

$$\deg(A) = \tilde{O}(d^2)$$
$$\deg(B) = O(d)$$

We also give the proof of rationality of P(t) by first principles.

Zuniga-Galindo'03 gave a det poly-time algo when f completely splits over rationals

Our algo makes no such assumptions also works for f is defined over $\mathbb{Z}_p$.

A closed form expression for $N_k(f)$ for univariate f.

Corollary: If f is radical, $N_k(f)$ is constant for large enough k.

Proof Idea

Proof Idea

Difficulty in computing the poincare series

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \ + \cdots \ \infty$$

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \; + \cdots \;\; \infty$$

Definition of $N_k(f)$ is implicit.

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \; + \cdots \;\; \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \; + \cdots \;\; \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \ + \cdots \ \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \; + \cdots \;\; \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

$$P(t) = 1 + \frac{t}{p} + \frac{t^2}{p^2} + \cdots \infty$$

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \; + \cdots \;\; \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

$$P(t) = 1 + \frac{t}{p} \; + \frac{t^2}{p^2} \; + \cdots \;\; \infty$$

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\,t + \frac{N_2(f)}{p^2}\,t^2 + \cdots \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

$$P(t) = 1 + \frac{t}{p} + \frac{t^2}{p^2} + \cdots \infty \quad \longrightarrow \quad P(t) = \frac{1}{1 - t/p}$$

Proof Idea

Difficulty in computing the poincare series

$$\boxed{P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \ + \cdots \ \infty}$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

$$\boxed{P(t) = 1 + \frac{t}{p} \ + \frac{t^2}{p^2} \ + \cdots \ \infty} \longrightarrow \boxed{P(t) = \frac{1}{1 - t/p}}$$

Our proof goes via computing closed form expression for $N_k(f)$ when $k>=k_0$.

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\,t + \frac{N_2(f)}{p^2}\,t^2 + \cdots \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

$$P(t) = 1 + \frac{t}{p} + \frac{t^2}{p^2} + \cdots \infty \quad \longrightarrow \quad P(t) = \frac{1}{1 - t/p}$$

Our proof goes via computing closed form expression for $N_k(f)$ when $k>=k_0$.

$$N_k(f) = \sum_{i=1}^{n} p^{k-\lceil \frac{k-v_i}{e_i} \rceil}$$

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \; + \cdots \; \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

$$P(t) = 1 + \frac{t}{p} + \frac{t^2}{p^2} + \cdots \; \infty \quad \longrightarrow \quad P(t) = \frac{1}{1 - t/p}$$

Our proof goes via computing closed form expression for $N_k(f)$ when $k>=k_0$.

$$N_k(f) = \sum_{i=1}^{n} p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$$

n = # distinct $\mathbb{Z}_p$ roots of f.

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \, + \cdots \, \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

$$P(t) = 1 + \frac{t}{p} + \frac{t^2}{p^2} + \cdots \, \infty \quad \longrightarrow \quad P(t) = \frac{1}{1 - t/p}$$

Our proof goes via computing closed form expression for $N_k(f)$ when $k>=k_0$.

$$N_k(f) = \sum_{i=1}^{n} p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$$

n = # distinct $\mathbb{Z}_p$ roots of f. e_i = multiplicity,

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 \; + \cdots \; \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

$$P(t) = 1 + \frac{t}{p} \; + \frac{t^2}{p^2} \; + \cdots \; \infty \quad \longrightarrow \quad P(t) = \frac{1}{1 - t/p}$$

Our proof goes via computing closed form expression for $N_k(f)$ when $k>=k_0$.

$$N_k(f) = \sum_{i=1}^{n} p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$$

n = # distinct $\mathbb{Z}_p$ roots of f. e_i = multiplicity, and v_i = constant

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough.

Too many terms!

What if $N_k(f)$ has nice explicit expression?

Say $N_k(f) = p^{k-1}$?

$$P(t) = 1 + \frac{t}{p} + \frac{t^2}{p^2} + \cdots \infty \quad \rightarrow \quad P(t) = \frac{1}{1 - t/p}$$

Our proof goes via computing closed form expression for $N_k(f)$ when $k>=k_0$.

$$N_k(f) = \sum_{i=1}^{n} p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$$

n = # distinct $\mathbb{Z}_p$ roots of f. e_i = multiplicity, and v_i = constant for i-th $\mathbb{Z}_p$ root.

Proof Idea

Difficulty in computing the poincare series

$$P(t) = N_0(f) + \frac{N_1(f)}{p}\, t + \frac{N_2(f)}{p^2}\, t^2 + \cdots \infty$$

Definition of $N_k(f)$ is implicit.

[DMS19] algo for computing $N_k(f)$ is not enough. Too many terms!

What if $N_k(f)$ has nice explicit expression? Say $N_k(f) = p^{k-1}$?

$$P(t) = 1 + \frac{t}{p} + \frac{t^2}{p^2} + \cdots \infty \;\longrightarrow\; P(t) = \frac{1}{1 - t/p}$$

Our proof goes via computing closed form expression for $N_k(f)$ when $k >= k_0$.

$$N_k(f) = \sum_{i=1}^{n} p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$$

Radical f: $$N_k(f) = \sum_{i=1}^{n} p^{v_i}$$

n = # distinct $\mathbb{Z}_p$ roots of f. e_i = multiplicity, and v_i = constant for i-th $\mathbb{Z}_p$ root.

Proof Idea

Proof Idea

Number of roots can be exponential.

Proof Idea

Number of roots can be exponential.

But each root is close to unique $\mathbb{Z}_p$ root.

Proof Idea

Number of roots can be exponential.

But each root is close to unique $\mathbb{Z}_p$ root.

Neighbourhood set of i-th root:

Proof Idea

Number of roots can be exponential.

But each root is close to unique $\mathbb{Z}_p$ root.

$\boxed{\text{Neighbourhood set of } i\text{-th root:}}$

Compact size $N_{k,i}(f) = p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$.

Proof Idea

Number of roots can be exponential.

But each root is close to unique $\mathbb{Z}_p$ root.

$\boxed{\text{Neighbourhood set of } i\text{-th root:}}$

Compact size $N_{k,i}(f) = p^{k-\lceil \frac{k-v_i}{e_i} \rceil}$.

So one can split up the Poincare series into n many series

Proof Idea

Number of roots can be exponential.

But each root is close to unique $\mathbb{Z}_p$ root.

Neighbourhood set of i-th root:

Compact size $N_{k,i}(f) = p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$.

So one can split up the Poincare series into n many series

$$P_i(t) = \frac{N_{k_0,i}(f)}{p^{k_0}} t^{k_0} + \frac{N_{k_0+1,i}(f)}{p^{k_0+1}} t^{k_0+1} + \frac{N_{k_0+2,i}(f)}{p^{k_0+2}} t^{k_0+2} + \cdots \infty$$

Proof Idea

Number of roots can be exponential.

But each root is close to unique $\mathbb{Z}_p$ root.

$\boxed{\text{Neighbourhood set of i-th root:}}$

Compact size $N_{k,i}(f) = p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$.

So one can split up the Poincare series into n many series

$$\boxed{P_i(t) = \frac{N_{k_0,i}(f)}{p^{k_0}} t^{k_0} + \frac{N_{k_0+1,i}(f)}{p^{k_0+1}} t^{k_0+1} + \frac{N_{k_0+2,i}(f)}{p^{k_0+2}} t^{k_0+2} + \cdots \infty}$$

$N_{k,i}(f)$ is a nice p-power with exponent linear in k we get $P_i(t)$ in terms of vi and ei.

Proof Idea

Number of roots can be exponential.

But each root is close to unique $\mathbb{Z}_p$ root.

$\boxed{\text{Neighbourhood set of i-th root:}}$

Compact size $N_{k,i}(f) = p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$.

So one can split up the Poincare series into n many series

$$\boxed{P_i(t) = \frac{N_{k_0,i}(f)}{p^{k_0}} t^{k_0} + \frac{N_{k_0+1,i}(f)}{p^{k_0+1}} t^{k_0+1} + \frac{N_{k_0+2,i}(f)}{p^{k_0+2}} t^{k_0+2} + \cdots \infty}$$

$N_{k,i}(f)$ is a nice p-power with exponent linear in k we get $P_i(t)$ in terms of v_i and e_i.

$$\boxed{P_i(t) = \frac{t^{k_i}(p - t(p-1) - t^{e_i})}{p^{(k_i - v_i)/e_i}(1-t)(p - t^{e_i})}}$$

Proof Idea

Number of roots can be exponential.

But each root is close to unique $\mathbb{Z}_p$ root.

Neighbourhood set of i-th root:

Compact size $N_{k,i}(f) = p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$.

So one can split up the Poincare series into n many series

$$P_i(t) = \frac{N_{k_0,i}(f)}{p^{k_0}} t^{k_0} + \frac{N_{k_0+1,i}(f)}{p^{k_0+1}} t^{k_0+1} + \frac{N_{k_0+2,i}(f)}{p^{k_0+2}} t^{k_0+2} + \cdots \infty$$

$N_{k,i}(f)$ is a nice p-power with exponent linear in k we get $P_i(t)$ in terms of v_i and e_i.

$$P_i(t) = \frac{t^{k_i}(p - t(p-1) - t^{e_i})}{p^{(k_i - v_i)/e_i}(1-t)(p - t^{e_i})}$$

$N_0(f), N_1(f), \ldots, N_{k_0-1}(f)$ are computed by calling [DMS19] k_0 times.

Proof Idea

Number of roots can be exponential.

But each root is close to unique $\mathbb{Z}_p$ root.

$\boxed{\text{Neighbourhood set of } i\text{-th root:}}$

Compact size $N_{k,i}(f) = p^{k-\lceil \frac{k-v_i}{e_i} \rceil}$.

So one can split up the Poincare series into n many series

$$\boxed{P_i(t) = \frac{N_{k_0,i}(f)}{p^{k_0}} t^{k_0} + \frac{N_{k_0+1,i}(f)}{p^{k_0+1}} t^{k_0+1} + \frac{N_{k_0+2,i}(f)}{p^{k_0+2}} t^{k_0+2} + \cdots \infty}$$

$N_{k,i}(f)$ is a nice p-power with exponent linear in k we get $P_i(t)$ in terms of v_i and e_i.

$$\boxed{P_i(t) = \frac{t^{k_i}(p - t(p-1) - t^{e_i})}{p^{(k_i - v_i)/e_i}(1-t)(p - t^{e_i})}}$$

$N_0(f), N_1(f), \ldots, N_{k_0-1}(f)$ are computed by calling [DMS19] k_0 times. $\boxed{k_0 = \tilde{O}(d^2).}$

Proof Idea

Number of roots can be exponential. But each root is close to unique $\mathbb{Z}_p$ root.

$\boxed{\text{Neighbourhood set of } i\text{-th root:}}$ Compact size $N_{k,i}(f) = p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$.

So one can split up the Poincare series into n many series

$$\boxed{P_i(t) = \frac{N_{k_0,i}(f)}{p^{k_0}} t^{k_0} + \frac{N_{k_0+1,i}(f)}{p^{k_0+1}} t^{k_0+1} + \frac{N_{k_0+2,i}(f)}{p^{k_0+2}} t^{k_0+2} + \cdots \infty}$$

$N_{k,i}(f)$ is a nice p-power with exponent linear in k we get $P_i(t)$ in terms of v_i and e_i.

$$\boxed{P_i(t) = \frac{t^{k_i}(p - t(p-1) - t^{e_i})}{p^{(k_i - v_i)/e_i}(1-t)(p - t^{e_i})}}$$

$N_0(f), N_1(f), \ldots, N_{k_0-1}(f)$ are computed by calling [DMS19] k_0 times. $\boxed{k_0 = \tilde{O}(d^2).}$

Use [DMS19] to compute $N_{k,i}(f)$ and equate with its expression.

Proof Idea

Number of roots can be exponential. But each root is close to unique $\mathbb{Z}_p$ root.

Neighbourhood set of i-th root: Compact size $N_{k,i}(f) = p^{k - \lceil \frac{k - v_i}{e_i} \rceil}$.

So one can split up the Poincare series into n many series

$$P_i(t) = \frac{N_{k_0,i}(f)}{p^{k_0}} t^{k_0} + \frac{N_{k_0+1,i}(f)}{p^{k_0+1}} t^{k_0+1} + \frac{N_{k_0+2,i}(f)}{p^{k_0+2}} t^{k_0+2} + \cdots \infty$$

$N_{k,i}(f)$ is a nice p-power with exponent linear in k we get $P_i(t)$ in terms of v_i and e_i.

$$P_i(t) = \frac{t^{k_i}(p - t(p-1) - t^{e_i})}{p^{(k_i - v_i)/e_i}(1-t)(p - t^{e_i})}$$

$N_0(f), N_1(f), \ldots, N_{k_0-1}(f)$ are computed by calling [DMS19] k_0 times. $k_0 = \tilde{O}(d^2)$.

Use [DMS19] to compute $N_{k,i}(f)$ and equate with its expression. Get v_i and e_i.

Conclusion

Conclusion

We gave an algorithm to compute Igusa local zeta function (Poincare series)

Conclusion

We gave an algorithm to compute Igusa local zeta function (Poincare series) associated to a univariate polynomial which runs in deterministic poly-time.

Conclusion

We gave an algorithm to compute Igusa local zeta function (Poincare series) associated to a univariate polynomial which runs in deterministic poly-time.

Could we generalise this to n-variable polynomials?

Conclusion

We gave an algorithm to compute Igusa local zeta function (Poincare series) associated to a univariate polynomial which runs in deterministic poly-time.

Could we generalise this to n-variable polynomials? Say n=2?

Conclusion

We gave an algorithm to compute Igusa local zeta function (Poincare series) associated to a univariate polynomial which runs in deterministic poly-time.

Could we generalise this to n-variable polynomials? Say n=2?

For a general n-variate polynomial even root-counting is NP-Hard!

Conclusion

We gave an algorithm to compute Igusa local zeta function (Poincare series) associated to a univariate polynomial which runs in deterministic poly-time.

Could we generalise this to n-variable polynomials? Say n=2?

For a general n-variate polynomial even root-counting is NP-Hard!

Thanks for your attention.